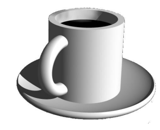
Programmation Orientée Objet - Licence TIS CM7/9

Lancelot Pecquet
Lancelot.Pecquet@math.univ-poitiers.fr



Poitiers, le 06/03/2006

UML
Conception OO
Motifs de conception (design patterns)
Projet

Rappel sur la séance précédente

La fois précédente, nous avons vu :

1. transtypage
2. égalité
3. comparabilité
4. clonage

Aujourd'hui, nous voyons :

Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Conception OO

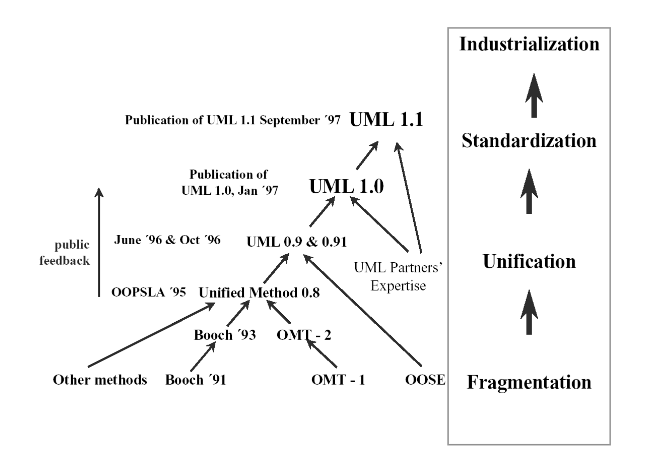
Caractéristiques d'un objet

- identité
- état interne (encapsulation)
- comportement vis à vis de l'extérieur (passage de messages)

UML

- UML = *Unified Modeling Language*
  = (Langage de Modélisation Unifié)
- langage de description graphique OO
- facilite l'expression et la communication des modèles
- indépendant du langage OO
- n'est pas une *méthode* de développement

Historique d'UML (V 2.0 en cours de standardisation)



Diagrammes de classes

Définition

Une classe est la description d'un ensemble d'objets partageant les mêmes attributs, opérations, relations, sémantique.

Représentation graphique d'une classe

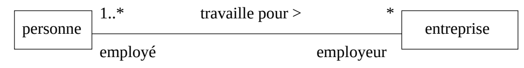
- un nom (*e.g.* `Forme`, voire `Geom : :Forme`)
- 0 à *n* attributs (champs) : nom (+ évt. type et valeur)
- 0 à *n* opérations (méthodes) : nom (+ évt. *profil* – prototype – et valeur)
- une *responsabilité* (description informelle)
- l'important est la lisibilité : tout n'a pas à être explicité.

Principaux types d'opérations

- constructeurs
- destructeurs
- sélecteurs (accesseurs)
- modifieurs (mutateurs)
- itérateurs (énumération des composants)

Relations entre les classes

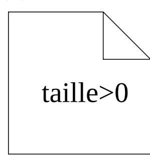
- associations : connexion sans flèche avec nom suivi d'un triangle pointant vers le "complément d'objet" + + nb d'instances concernées :



- association (partie d'un tout) : flèche à pointe losange creux côté tout (*e.g.* mots / phrase)
- composition (aggrégation forte) : flèche à pointe losange noire côté tout (*e.g.* bras / homme)
- généralisation (héritage) : flèche à extrêmité triangulaire (on n'écrit que les nouveaux attributs et nouvelles opérations)

Notes et contraintes

- un cadre avec l'angle droit corné permet d'indiquer une note sur une classe ou une relation :

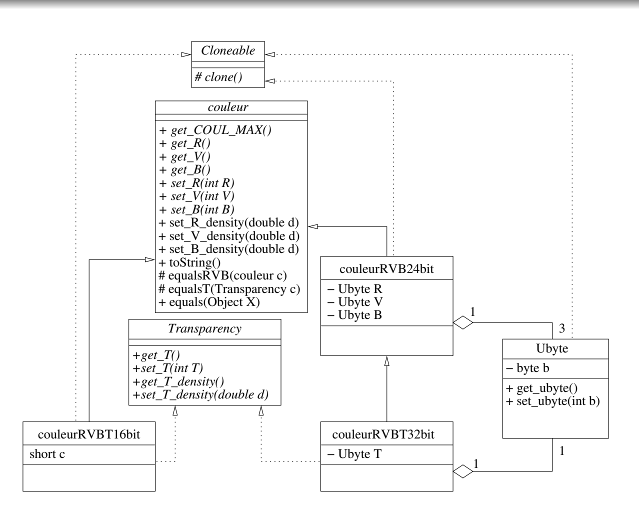


- un commentaire peut également être ajouté entre double chevrons (*e.g.* ⟨⟨ mutateur ⟩⟩)

Autres éléments de langage UML

- au niveau de l'encapsulation
  - + désigne une encapsulation `public`
  - # désigne une encapsulation `protected`
  - − désigne une encapsulation `private`
- les interfaces sont indiquées en pointillé.

Exemple de diagramme de classes UML



UML
Conception OO
Motifs de conception (design patterns)
Projet
Idée de base d'UML
Classes
Références UML
Autres diagrammes UML

Références UML

- pour les cours :
  - *Modélisation objet avec UML*, Muller & Gartner, Eyrolles
  - `http://laurent-piechocki.developpez.com/uml/`
- pour les logiciels
  - Borland Together (Java, Delphi, C, C++...) ;
  - IBM : Rational Rose ;
  - Gentleware Poseidon for UML : `http://www.gentleware.com` (gratuit)

Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Diagrammes UML statiques

1. de cas d'utilisation : point de vue externe (acteur)
2. de classes : interaction structurelle entre les classes
3. d'objets : affine le précédent pour des instances données
4. de composants : interaction des éléments logiciels (exécutables, bibliothèques, fichiers. . .)
5. de déploiement : description physique des matériels

UML
Conception OO
Motifs de conception (design patterns)
Projet
Idée de base d'UML
Classes
Références UML
Autres diagrammes UML

Diagrammes UML dynamiques

1. d'états/transitions : cycle de vie des objets d'une classe
2. de collaboration : organisation des messages entre les objets
3. de séquence : chronologie de ces messages
4. d'activités : déroulement d'un processus (*e.g.* métier)

Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Diagramme de cas d'utilisation

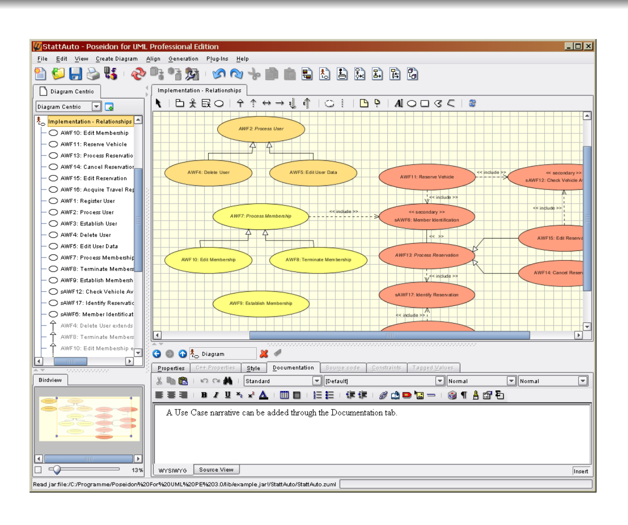
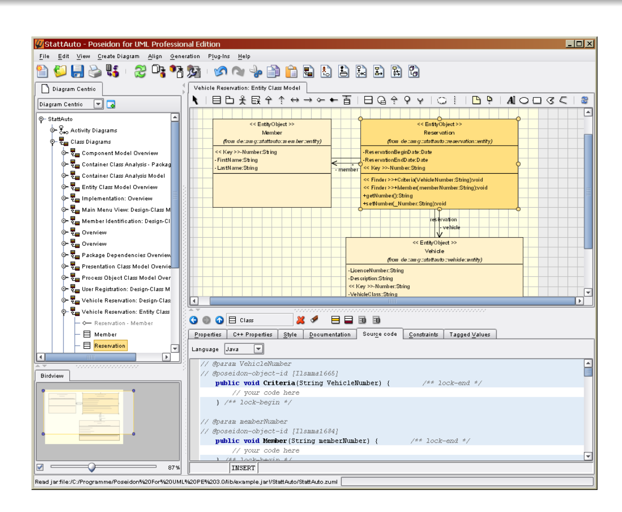


Diagramme de classes



Génération automatique de code Java

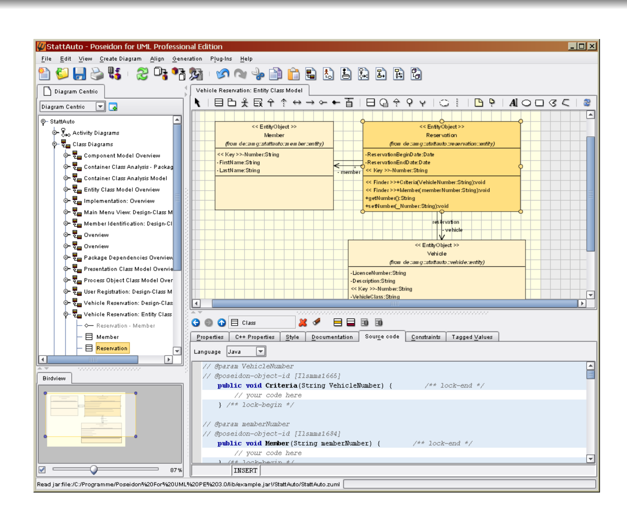


Diagramme d'objets

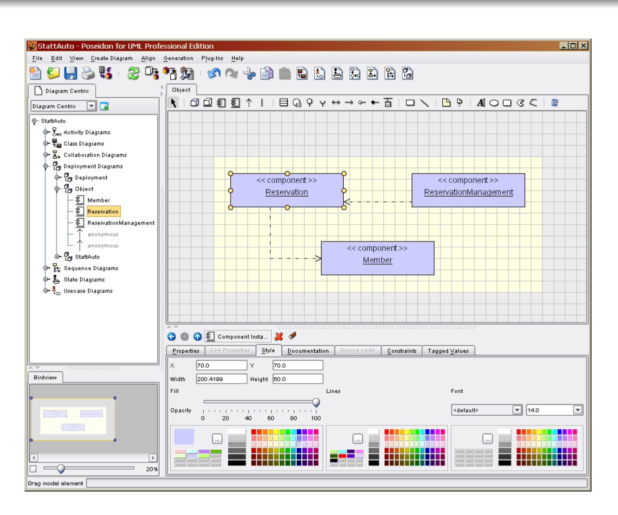


Diagramme d'états-transitions

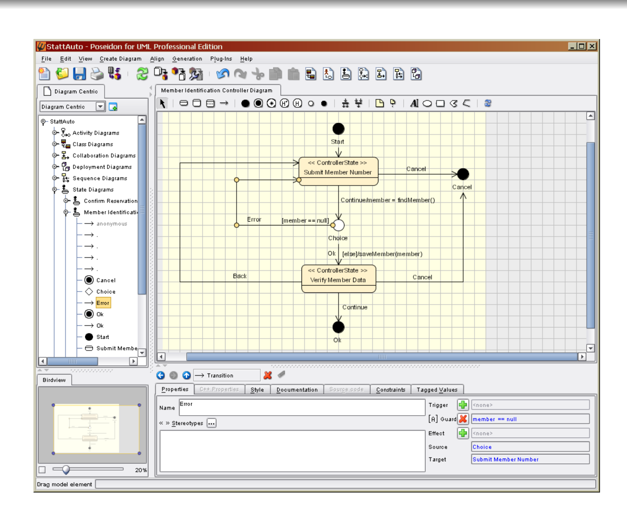


Diagramme de déploiement

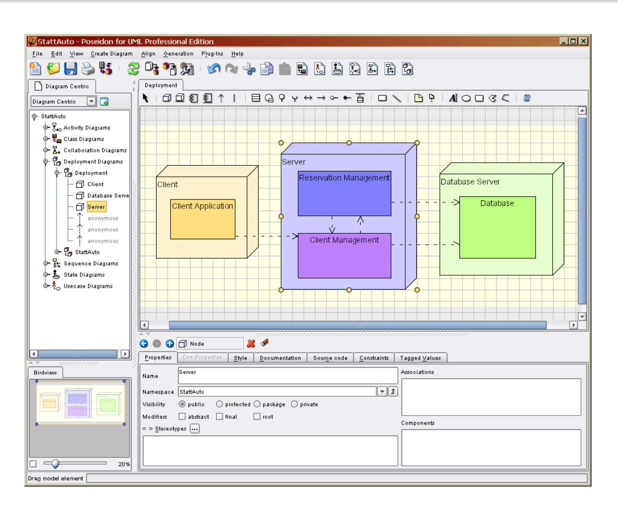


Diagramme de collaboration

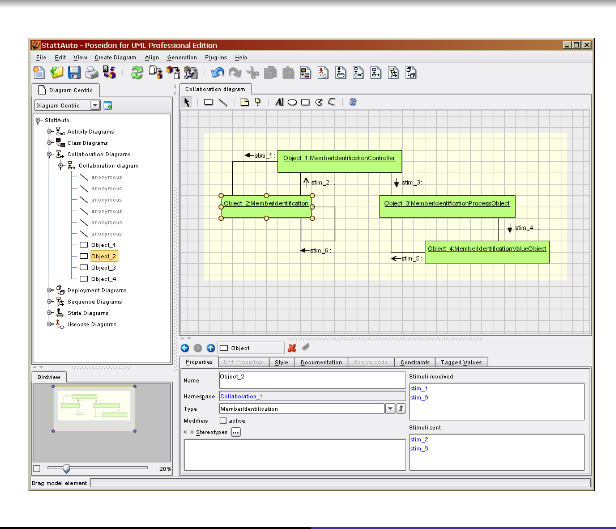


Diagramme de séquence

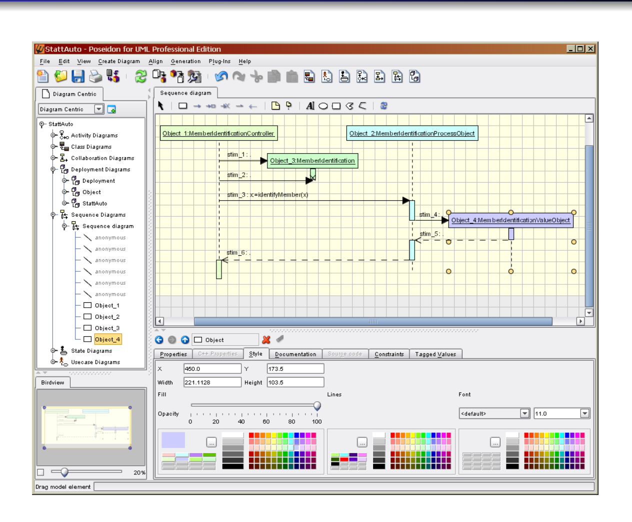
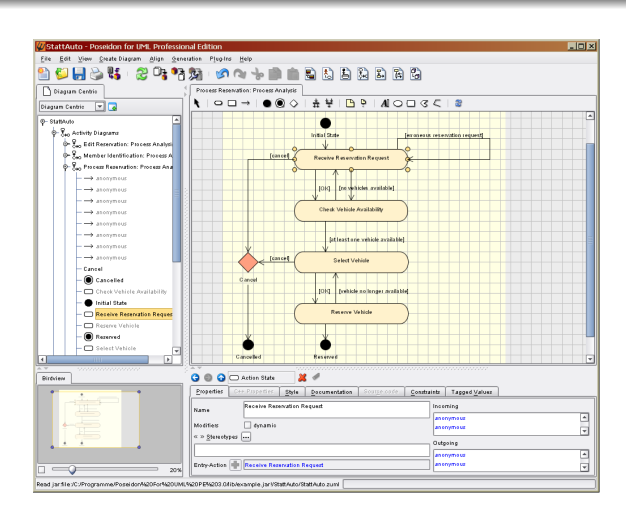
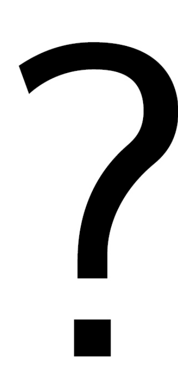
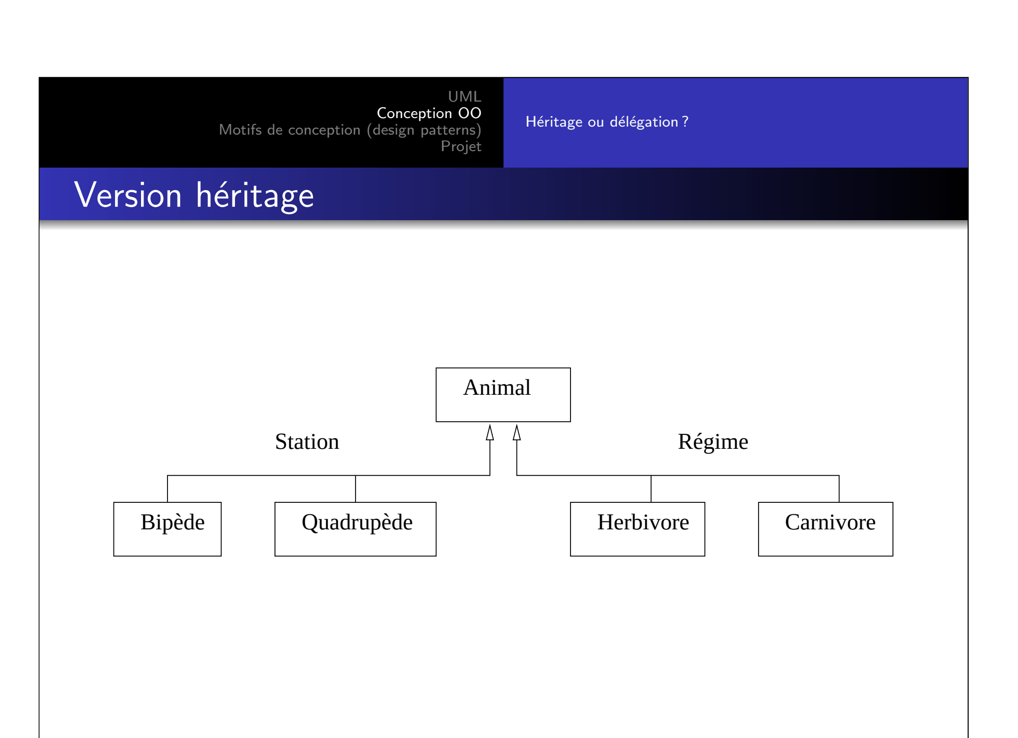


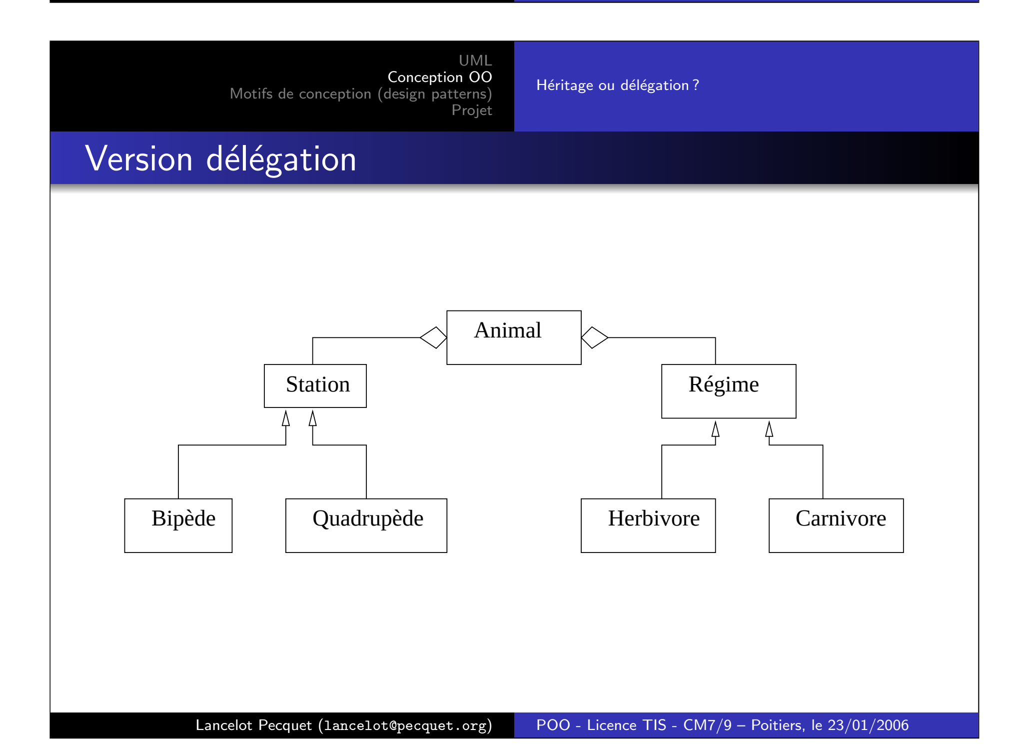
Diagramme d'activité



Questions ?




UML
Conception OO
Motifs de conception (design patterns)
Projet
Héritage ou délégation ?
Version héritage
Animal
Station
Régime
Bipède
Quadrupède
Herbivore
Carnivore
Lancelot Pecquet (lancelot@pecquet.org)
POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006


UML
Conception OO
Motifs de conception (design patterns)
Projet
Héritage ou délégation ?
Version délégation
Animal
Station
Régime
Bipède
Quadrupède
Herbivore
Carnivore
Lancelot Pecquet (lancelot@pecquet.org)
POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Héritage ou délégation ?

- héritage
  - attributs propagés automatiquement
  - polymorphisme
  - rigide
- délégation
  - attributs propagés au cas par cas
  - pas de polymorphisme
  - souple

Questions ?



Motifs de conception (*design patterns*)

Définition

Un motif de conception(*design pattern*) est une architecture de classes qui est solution d'un problème typique.

Pourquoi ?

- pour ne pas réinventer le fil à couper le beurre à chaque programme
- pour s'appuyer sur une conception fiable
- pour faciliter la collaboration entre programmeurs

Les différents styles de motifs du GOF

Les motifs du « Gang of Four (GOF) » (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) :

- motif créationnels (*creational patterns*)
- motif structurels (*structural patterns*)
- motif comportementaux (*behavioral patterns*)

Motifs créationnels (creational patterns)

- *Abstract Factory* : crèe une instance de plusieurs familles de classe
- *Builder* : sépare un objet de sa représentation
- *Factory Method* : crèe une instance de plusieurs classes dérivées
- *Prototype* : une instance pleinement initialisée destinée à la copie ou au clonage
- *Singleton* : une classe dans laquelle une seule instance est prévue

UML
Conception OO
Motifs de conception (design patterns)
Projet
Motifs créationnels (creational patterns)
Motifs structurels (structural patterns)
Motifs comportementaux (behavioural patterns)

Ex : Singleton

Singleton
-instance : Singleton
-Singleton() +Instance() : Singleton

```
public class Singleton {
   // Attention au parallelisme...
   private static Singleton instance = null;

   private Singleton() {
      // Redefini prive pour eviter d'autres instanciation
   }

   public static Singleton Instance() {
      if(instance == null) {
         instance = new Singleton();
      }
      return instance;
   }
}
```

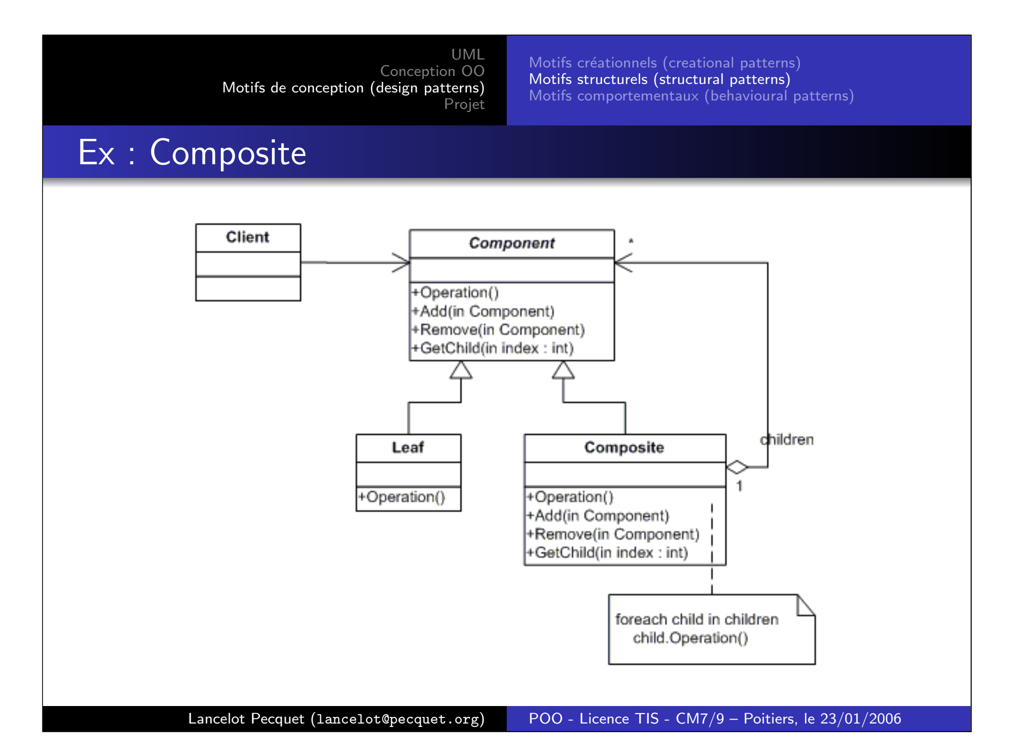
Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Motifs structurels (structural patterns)

- *Adapter* : adapte les interfaces de différentes classes
- *Bridge* : sépare les objets de leurs implémentations
- *Composite* : un arbre d'objets simples et composites
- *Decorator* : ajoute dynamiquement des responsabilités aux objets
- *Façade* : une seule classe représente un sous-système complet
- *Flyweight* : une instance "à fine granularité" est utilisée pour un partage efficace
- *Proxy* : un objet représente un autre objet

UML
Conception OO
Motifs de conception (design patterns)
Projet
Motifs créationnels (creational patterns)
Motifs structurels (structural patterns)
Motifs comportementaux (behavioural patterns)

Ex : Composite



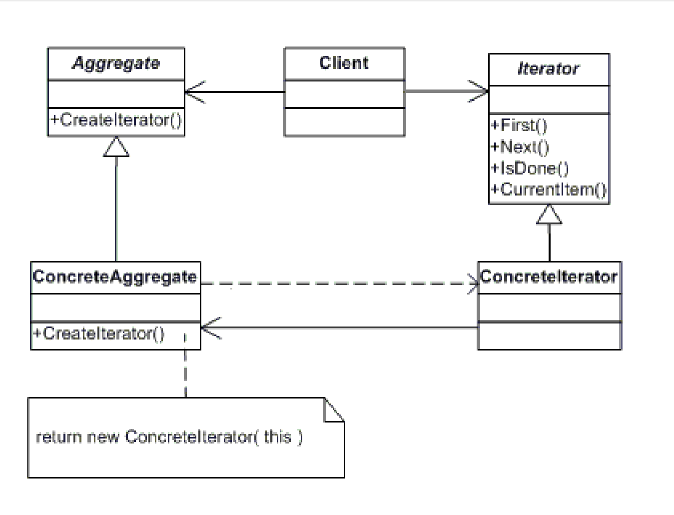
Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006

Motifs comportementaux (behavioural patterns)

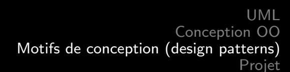
- *Chain of Resp.* : passage d'une requête à travers une suite d'objets
- *Command* : encapsule une requête de commande comme un objet
- *Interpreter* : inclusion d'éléments du langage dans un programme
- *Iterator* : une manière d'énumérer les éléments d'une collection
- *Mediator* : définit une communication simplifiée entre les classes
- *Memento* : capture et restore l'état interne d'un objet
- *Observer* : une façon de notifier un changement à des classes
- *State* : altère le comportement d'un objet lorsque son état change
- *Strategy* : encapsule un algorithme dans une classe
- *Template Method* : délègue le détail d'un algorithme à une sous-classe
- *Visitor* : définit une nouvelle opération pour une classe sans changement

UML
Conception OO
Motifs de conception (design patterns)
Projet
Motifs créationnels (creational patterns)
Motifs structurels (structural patterns)
Motifs comportementaux (behavioural patterns)

Ex : Iterator



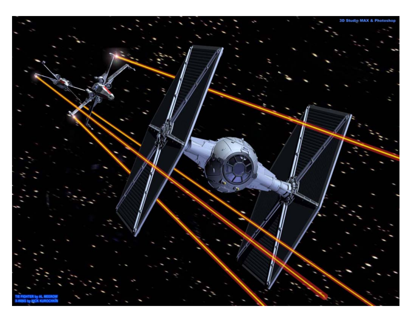
Lancelot Pecquet (lancelot@pecquet.org) POO - Licence TIS - CM7/9 – Poitiers, le 23/01/2006



Questions ?



Modélisation de bataille spatiale de Star Wars





Situation générale

- il y a deux flottes opposées : l'Alliance rebelle et l'Empire
- chaque vaisseau occupe une place sur une grille
- le but du jeu est d'éliminer la flotte ennemie

Situation générale

- lors d'un tour de jeu, chaque vaisseau peut :
  1. se déplacer de v unités (v est sa vitesse) ou moins dans n'importe quelle direction vers son ennemi le plus proche ou passer en hyperespace et se retrouver au hasard sur la grille
  2. si un ennemi est à portée de feu, utiliser une arme, blaster (dégâts : 1–5) ou missile (dégâts : 10), sur un vaisseau ennemi ; si aucun ennemi n'est à portée, il n'y a pas de tir
- un tir à $t \times 25\%$ de chances de réussir où t est la taille du vaisseau visé
- chaque point de bouclier permet d'absorber un point de dégâts ; un bouclier en dessous de zéro signifie que le vaisseau est détruit
- les tirs de blaster sont illimités, les tirs de missiles sont limités

X-Wing (Alliance rebelle)

- Nom : Chasseur T-65 X-Wing
- Constructeur : Incom Corporation
- Taille : 2
- Vitesse : 4/tour + hyperespace
- Attaque : 1 blaster de puissance 4 + 2 missiles
- Bouclier : 20



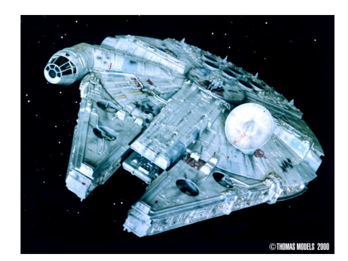
B-Wing (Alliance rebelle)

- Nom : Bombardier B-51 B-Wing
- Constructeur : Slayn et Korpil
- Taille : 2
- Vitesse : 4/tour + hyperespace
- Attaque : 1 blaster de puissance 3 + 16 missiles
- Bouclier : 20



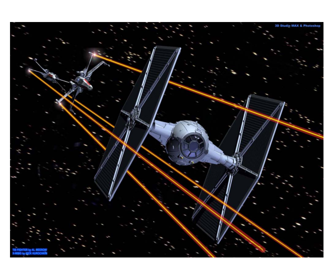
Faucon Millenium (Alliance rebelle)

- Nom : *Faucon Millenium* (un seul exemplaire)
- Constructeur : Corellian Tech. Corp.
- Taille : 3
- Vitesse : 6/tour + hyperespace
- Attaque : 1 blaster de puissance 5 + 16 missiles
- Bouclier : 30



Chasseur TIE (Twin Ion Engine, Empire)

- Nom : Chasseur TIE
- Constructeur : Sienar Fleet Systems
- Taille : 1
- Vitesse : 6/tour
- Attaque : 1 blaster de puissance 2
- Bouclier : 0



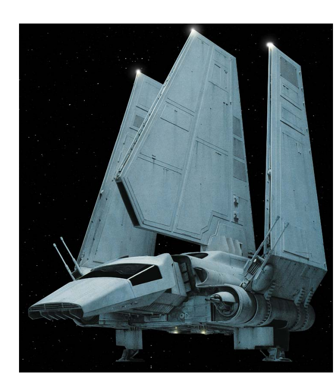
Bombardier TIE (Empire)

- Nom : Bombardier TIE
- Constructeur : Sienar Fleet Systems
- Taille : 2
- Vitesse : 4/tour
- Attaque : 1 blaster de puissance 1 + 30 missiles
- Bouclier : 0



Navette de classe lambda (Empire)

- Nom : Navette de classe Lambda
- Constructeur : Sienar Fleet Systems
- Taille : 2
- Vitesse : 3/tour + hyperespace
- Attaque : 1 blaster de puissance 5
- Bouclier : 25



Objectifs principaux

- diagramme de classes UML
- composants : armes, vaisseaux, situation de jeu
- destruction des vaisseaux avec des exceptions
- redéfinir la performance des vaisseaux si pilote Jedi ou Sith
- tests → situations jouables ?

Options

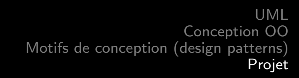
- sauvegardes
- interface graphique
- jeu en 3D
- son
- autres armes (canons à ions, torpilles à protons...) et vaisseaux (A-wing, Y-wing, intercepteur TIE, destroyer impériaux...)

Conseils : à chaque étape :

- sauvegarder et conserver les versions successives : il faut toujours avoir une version qui marche
- définir le plus tôt possible un affichage clair et simple
- noter dans votre rapport les éléments algorithmiques, techniques et difficultés rencontrées
- de commenter les classes et de générer une documentation `javadoc` pour mieux vous y retrouver et améliorer la rédaction de votre rapport

Évaluation

- Archive `.zip` contenant :
  - le programme commenté (coeff. 4)
  - la doc `javadoc`
  - un rapport en PDF (10pp. *maximum* – coeff. 1)
  - un mode d'emploi en PDF (1p. *maximum* – coeff. 1)
- **Soutenance** (coeff. 2) : 10 min + 5 min de questions.



Questions ?



Conclusion

Aujourd'hui, nous avons vu :

1. UML
2. Conception OO
3. Motifs de conception (design patterns)
4. Projet

La séance prochaine, nous verrons :

1. fichiers
2. sérialisation
3. threads
4. applets